An Algorithm Using Length-R Paths to Approximate Subgraph Isomorphism

Fred DePiero* **and David Krout**
Cal Poly State University

Abstract The 'LeRP' algorithm approximates subgraph isomorphism for attributed graphs based on counts of Length-R Paths. The algorithm provides a good approximation to the maximal isomorphic subgraph. The basic approach of the LeRP algorithm differs fundamentally from other methods. When comparing structural similarity LeRP uses a neighborhood of nodes that varies in size dynamically. This approach provides sufficient evidence of similarity to permit LeRP to form a node-to-node mapping and can be computed with polynomial effort in the worst-case. Results from over 32,000 simulated cases are reported. We demonstrate that LeRP does not need a high dynamic range of node and edge coloring to perform well. For example, LeRP can input 50-node and 100-node graphs that contain a common 50-node subgraph, and then compute a matching subgraph having 49.74 +/- 0.46 nodes (mean +/- one standard deviation). This takes from 0.4 to 0.5 seconds. In this example, 100 trials were evaluated and graphs had discrete coloring for nodes and edges with a dynamic range of four. Test conditions are varied and include strongly regular graphs as well as Model A.

Keywords: Approximate Graph Matching, Subgraph Isomorphism, Attributed Relational Graphs

Introduction and Problem Definition

A graph **G** is defined to have a set of $\mathbf{N_G}$ nodes, $\mathbf{g_i}$, and **Q** edges, $\mathbf{e_{ij}}$. We assume that graphs are not directed. Nodes and edges are permitted to have coloring. But, LeRP does not require this coloring, or any variation in degree, in order to function. We do not address multigraphs. However, this is not regarded as a restriction because a representation could be created to describe the existence of parallel edges via an edge-coloring scheme.

Graphs **G** and **H** (having nodes $\mathbf{g_i}$ and $\mathbf{h_k}$, respectively) are isomorphic if there exists a mapping, $\mathbf{h_k = m(g_i)}$, that preserves the adjacency of all nodes. Any application-specific node and edge colorings must also be consistent under the mapping. Subgraph isomorphism is a condition of isomorphism that exists between **G** and a subgraph of **H**.

The adjacency matrix **A** of graph **G** plays an important role in verifying isomorphism [31]. Given a node-to-node mapping, **m()**, between two graphs **G** and **H**, isomorphism may be verified by reordering the nodes and relocating the edges of **H**. The reordered adjacency matrix **B'** should then equal **A** – if an isomorphism exists. This eliminates the possibility of a false-positive result.

An automorphism is an isomorphic mapping of the nodes of **G** onto itself. Automorphisms arise due to symmetries in graph structures [31]. These are significant because the symmetries can result in multiple solutions to the isomorphism problem. In these cases two different mappings will result in identical adjacency matrices. We consider both to be valid, provided node and edge colorings are also in agreement.

Challenge of Determining Graph and Subgraph Isomorphism

The amount of effort associated with a brute force solution would typically be considered intractable, as there are **N!** possible node orders for a graph with **N** nodes. The subgraph isomorphism problem is even worse combinatorically, as the size of the matching subgraph is unknown. The complexity class for the graph isomorphism problem is not currently known [32], but the subgraph isomorphism problem is known to be NP-Complete.

A factor increasing the difficulty of the isomorphism problem can be the (lack of) variation in degree. The degree forms a natural classification for nodes, permitting candidate node mappings to be pruned out. Note

* Author for all correspondence.

however, that opportunities for this type of pruning may be limited in practice. The dynamic range of node and edge coloring is another important factor that affects the computational demand and the accuracy of matching algorithms. Higher dynamic ranges greatly reduce potential matches.

Algorithm Development Goals for LeRP

Our goal for this algorithm was to create an approximate technique that can identify near-maximal subgraphs with a polynomial bound on (the total) processing effort. It is due to the NP-Completeness of the subgraph isomorphism problem that an approximate solution was sought. Memory requirements of a low order were also regarded as important.

The criteria we use to judge an acceptable algorithm is the ability to maintain a near-maximal matched subgraph, while requiring a low dynamic range of node and edge coloring. The low dynamic range means that the matching process does not rely on a high degree of distinction in node and edge coloring. In this situation, matching is achieved, thanks to structural comparisons. This makes for an algorithm that may have utility in a broad number of applications. We also strove for a technique that could operate on generic styles of graphs, rather than graphs with some particular structure.

Horizon of Node-to-Node Comparisons – A Taxonomy

We assess the structural similarity of a pair of nodes, $\mathbf{g_i}$ and $\mathbf{h_k}$, during the process of evaluating candidate mappings. The "horizon" associated with this comparison is defined as the furthest distance away from $\mathbf{g_i}$ (or from $\mathbf{h_k}$) that data is considered in the assessment.

A shorter horizon may involve less computation for the comparison itself, but can mislead the matching process because comparisons are more local in nature. This can result in the need to backtrack during the matching process. Backtracking occurs when a pairing that seems initially acceptable is later found to be incorrect [9]. As a result of a limited horizon, isomorphism techniques often suffer as graphs become larger. So, if backtracking is to be avoided (or limited) then the comparison horizon will generally need to increase as **N** increases.

Note that some limitation on the comparison horizon is necessary for the subgraph-matching problem. By limiting the horizon, portions of the graph structure that differ won't corrupt the description of a node that is actually within part of a matching subgraph. Hence there is a tradeoff between overall computational effort and the accuracy of comparisons. This tradeoff is directly tied to the comparison horizon. Due to the impact of the horizon distance on algorithm performance, it can serve as an effective taxonomy for comparing alternate matching techniques.

LeRP Dynamic Comparison Horizon - Contrast to Other Approaches

Two broad categories of approaches have been studied in the machine vision community: exhaustive and approximate. Exhaustive techniques include [1] [13] [23] [19] [7] [3] and [14]. These types of methods typically use local comparisons of nodes (a short horizon) but then enforce consistencies from node-to-node, in a manner that eventually yields a globally optimum solution.

Exhaustive Techniques

An A* search [18] can be used to build an interpretation tree [9] that will enumerate all possible mappings in a depth-first order, for example. In this situation, the consistency checks at each level in the tree will be local in nature. However, following a path from the root of the tree down to a leaf results in a global check. The comparison horizon is forced to extend out to maximal size in this manner.

Another classic example is the search for a maximum sized clique within an association graph [1] [14]. Each node of the association graph describes a local comparison. The maximum clique identifies the largest possible subset of nodes (in the association graph) that is consistent. Here again local comparisons are used.

Then via an iterative process, a more global consistency in graph structure is found. The iterative process may end with either a matched graph, or subgraph, depending on the input conditions

Of course, exhaustive techniques that attempt to find globally optimal solutions suffer from the NP-Completeness of subgraph matching. This results in a worst-case complexity that is exponential in order. Hence the long term interest in approximate techniques [8] [24] [16] [4], which are adequate in many applications, and are also commensurate with the goals of this investigation. Approximate techniques may be deemed appropriate, for example, when graphs are constructed from noisy sensor data. In this situation the input graphs contain some noise or imperfect data. Hence the possibility of an ideal match is precluded.

Approximate Techniques

In relaxation techniques [24] [25] [10], all locally consistent assignments are made first (short horizon distance). Then the horizon is effectively increased via a constraint propagation that occurs in an iterative fashion. Earlier relaxation techniques did not refine probability estimates as iterations progressed. More recent work [27-30] does not suffer from this limitation.

Structural comparisons made via a graph edit distance [16] [11] make counts of missing edges or nodes via local comparisons. These differences are then summed over an entire graph. In relational indexing [4], occurrences of 2-node graphs are binned and accumulated in a voting scheme. Note the fixed comparison horizon. It is rigid, irrespective of the extent of any similar neighboring structure that might be present.

In the work reported by Wilson and Hancock [10], a fixed substructure is also used. This consists of the central node under consideration, its incident edges and adjacent nodes. While this is a larger immediate neighborhood than most techniques, it is still rigid in structure. This is in contrast to a more dynamic approach used in LeRP.

LeRP Dynamic Comparison Horizon

In a number of cases - with both exhaustive and approximate techniques - a comparison using a local horizon is extended by some means. This is typically done by either iteration or search. This is fundamentally different than in the LeRP approach. By comparing the length-r paths, or circuits, that are associated with a node, the comparison neighborhood can potentially be quite large – depending on the value of r.

Although the comparison horizon can become quite large, potentially including every node, the overall LeRP algorithm is still approximate in nature. There is because the length-r path counts are *related* to graph structure, but do not guarantee that identical structures are present.

Comparison of LeRP with Other Methods

We consider a dynamic horizon to be superior to a fixed horizon distance, set to some arbitrary value. In general, a dynamic horizon can include numerous ‘nearby’ nodes. This increases the evidential merit of an individual comparison. The dynamic nature of the horizon allows the comparison to include more nodes in regions of the graph that are farther away from structural differences.

The horizon distance is a useful way to compare fundamental operation of various techniques. For example with probabilistic relaxation the horizon distance is dynamic, but it increases in an iterative fashion. In LeRP, a node-to-node comparison with a maximum horizon distance is possible on the first pass through the nodes. In fact this sort of assignment is typically made for the nodes further away from structural differences. It is the nodes that are closer to the differences that require additional computational effort – to ‘fill out’ the mapping to a near maximal size.

In methods based on graph-edit distance [16][11] a sum of occurrences of local differences is accumulated for the whole graph. In relational indexing [4] and in the work of Wilson and Hancock [10] graph

substructures of fixed-size are used. In each of these methods the horizon does not extend in a dynamic fashion – based on the data – as is done in LeRP. The dynamic comparison horizon improves the accuracy of structural comparisons for LeRP. This is a fundamental benefit to the approach.

The improved method of structural comparisons results in a relatively low dependence on the dynamic range of node and edge coloring. Few published works provide extensive testing in this area, but the work of Messmer and Bunke [16] is a fortunate exception. Their algorithm for error-tolerant matching was optimized for different goals than LeRP. Nevertheless, it is interesting to note that their tests typically had a dynamic range for coloring of 25%-50% (expressing dynamic range as a percentage of the number of nodes). In trials where Messmer and Bunke study acceptable ranges of node coloring, they show results becoming rapidly worse with a dynamic range below 20%. (Their tests only use node coloring, so trials of this style have also been included, see Table 2.) We demonstrate acceptable results – maintaining above 99% of the maximum subgraph size – with a dynamic range as low as 10%, half that reported in [16]. When both node and edge coloring is used, the dynamic range can be reduced to 4%.

A low dependency on node and edge coloring is important for applications, as it can reduce problems with sensor noise. These problems result when sensor noise corrupts node and edge colors, making the node-to-node comparisons problematic. Node and edge colors may be derived from sensors having continuous signals (or near-continuous signals). To mitigate noise problems it is helpful to sometimes form coarse bins for the color values ('high', 'medium', or 'low'), or to first subtract the color values and then perform a similar quantization. In either case the dynamic range (of quantized colors) is reduced. This potentially hinders the subgraph matching process, but does mask the sensor noise. Hence matching algorithms that can operate with low dynamic ranges of coloring are quite useful in applications with noisy sensor data.

Another important capability for matching algorithms is deterministic processing. Generally speaking, approximate methods trade off maximally sized subgraphs for tractable processing. This has a great benefit for applications. However, the next area for improvement is greater determinism. This sets the course towards an eventual goal of real-time evaluation of subgraph isomorphisms. Some techniques report relatively fast computations [16]. Others report a low degree of iterations [8], as do Luo and Hancock [34] with convergence in 12 iterations using the EM algorithm. However, no methods reported to date describe fixed limits on processing and the determinism of the LeRP algorithm. For example, tests demonstrate that LeRP can match 100-node and 50-node graphs, containing a 50 node subgraph, and with a dynamic range of 4 for node and edge coloring, in 0.4 seconds with a standard deviation less than 0.1 seconds. This is approaching the needs for computational speed and determinism associated with applications of real-time machine vision.

Further Comparisons

In contrast to relaxation techniques, LeRP does not begin by first allowing all possible assignments (of model vs. scene, for example) and then iteratively remove impossible matches. Rather, LeRP builds the mapping from a single node-to-node assignment (a 'mapping seed'). It then adds to this mapping, employing a consistency check as nodes are added to the subgraph.

A number of common methods for object recognition combine the problems of recognition and pose determination. Methods such as local-feature-focus [2], pose clustering [21], and geometric hashing [22], for example. These methods are able to employ additional constraints, generally involving the physical location of graph nodes, to limit potential matches. Some methods of registration also use this type of approach [6]. While these application areas are of interest, this style of constraint was intentionally not utilized. Rather, a more generic approach was sought. Hence in LeRP, the comparisons of structural similarity are based purely on node adjacencies – not on the physical location of nodes.

Some methods of finding subgraph isomorphisms require a significant amount of memory [13]. We strove to keep the memory requirements on the order $\mathbf{N^2}$, in order to be able to handle larger problem sizes. Also, because of LeRP's ability to process subgraphs directly, no padding is necessary in order to make the number of nodes equal in the two input graphs, as in some other approaches [13] [10]. Also note that there is a speed vs. memory tradeoff that is possible in LeRP, whereby certain precomputed values are not stored

and rather computed (repeatedly) when needed. This reduces the needed memory. See section on Memory and Compute Bounds.

Computing the Structural Similarity of Nodes and Edges

The adjacency matrix, **A**, of graph **G** is $\mathbf{N_G x N_G}$ and plays a key role when computing the structural similarity of nodes. **A** contains a 1 at location **(i, j)** if edge $\mathbf{e_{ij}}$ exists and a 0 for no edge. The **(i, i)** elements of **A** are also set to 1 in the LeRP technique. **A** is symmetric because no digraphs are assumed. The **(i, j)** element of $\mathbf{A^r}$, $\mathbf{a_{ij}^{(r)}}$, gives the number of length-r paths between node $\mathbf{g_i}$ and $\mathbf{g_j}$. The **(i, i)** element of $\mathbf{A^r}$ gives the number of closed paths (circuits) that include node $\mathbf{g_i}$ [31]. Because $\mathbf{a_{ii}^{(1)} = 1}$, the number of paths and circuits in $\mathbf{A^r}$ include the effect of self loops that are connected to each node.

Note that the standard definition of **A** uses $\mathbf{a_{ii}^{(1)} = 0}$ [31]. While our definition of $\mathbf{A^r}$ does differ from the standard, it makes the $\mathbf{a_{ij}^{(r)}}$ elements more 'well behaved'. Specifically, the $\mathbf{a_{ij}^{(r)}}$ elements increase monotonically with increasing **r**. Empirically, this appears to yield improved results in terms of the size of the matched subgraph.

Given two nodes $\mathbf{g_i}$ and $\mathbf{h_k}$ from graphs **G** and **H**, the structural similarity is computed via the $\mathbf{a_{ij}^{(r)}}$ and $\mathbf{b_{kl}^{(r)}}$ elements of the r-power adjacency matrices $\mathbf{A^r}$ and $\mathbf{B^r}$. The similarity metric is based on the largest value of **r** such that $\mathbf{a_{ij}^{(r)} = b_{kl}^{(r)}}$. Note that **r** is limited to a peak value of **R**, which is a parameter that must be set for a given application. In all the 32,000 trials reported here, **R = 10**. In this notation, **i,j** are two nodes in **G**, and **k,l** are two nodes in **H**.

LeRP uses a function ***compare(i,j,k,l)*** to compute the similarity measure. See Figure 1. The function returns a value between 0 and 1, normalized by $\mathbf{(r_{max} / N)^2}$ where $\mathbf{r_{max}}$ is the largest value of r such that that $\mathbf{a_{ij}^{(r)} = b_{kl}^{(r)}}$. Alternate forms of this function have been studied [15].

In the ***compare()*** function $\mathbf{1 <= r_{max} <= R}$. The $\mathbf{r_{max}}$ value determines the horizon distance of a comparison. Hence the comparison horizon varies from being strictly local ($\mathbf{r_{max}=1}$) up to a value that could include every node in the graph ($\mathbf{r_{max}=R=N}$, and provided the graph is connected). This variation from local to global provides the dynamic horizon in LeRP.

The parameter **R** determines the highest power of the adjacency matrices **A** and **B** that must be computed. To reduce processing requirements, **R** is kept as low as possible. For the results reported herein **R=10** was adequate for graphs up through **N=400**. In general **R** needs to increase, as **N** increases, but it appears to be a weak function of **N.** No theoretical bounds are provided at this time.

```
Function: compare(i,j,k,l)
1. For 1<=r<=R
      a. If a_ij^(r) != b_kl^(r) Then Break
2. Next r
3. Return (r/N)^2
```

Figure 1. The compare() function described in pseudo-code. Note i,j are two nodes in G, and k,l are two nodes in H. And, if an edge, e_{ij}, exists between i,j then $a_{ij}^{(1)} = 1$. compare() approximates a probability value that describes similarity in graph structure.

The value returned by ***compare()*** will tend towards zero for dissimilar regions of the graph structure, and towards one for similar regions. As such, it is used as a very rough approximation to the probability that edges $\mathbf{e_{ij}}$ and $\mathbf{e_{kl}}$ are equivalent, in the sense of $\mathbf{h_k = m(g_i)}$ and $\mathbf{h_l = m(g_j)}$. The monotonically increasing property of the $\mathbf{a_{ij}^{(r)}}$ values makes for a better approximation to probability values than would be achieved via the standard definition of the adjacency matrix, which does not possess this property.

Combining Evidence of Structural Similarity

The measure of similarity found via the ***compare()*** function is used as evidence that two given nodes, $\mathbf{g_i}$ and $\mathbf{h_k}$, could form an isomorphic pairing – leading to the mapping $\mathbf{h_k = m(g_i)}$. Three different types of evidence are combined in LeRP, to form an overall measure of structural similarity. These three types of evidence are referred to as 'alpha', 'beta', and 'gamma'.

When comparing nodes $\mathbf{g_i}$ to $\mathbf{h_k}$, alpha-style evidence is computed via ***compare*(i,i,k,k)**. This is based on the number of length-r circuits that include these particular nodes. Beta-style evidence involves length-r path counts associated with edges that are incident to the $\mathbf{g_i}$ and $\mathbf{h_k}$ nodes. And, gamma-style evidence involves a comparison of the number of length-r circuits for nodes that are adjacent to $\mathbf{g_i}$ and $\mathbf{h_k}$.

Various methods to combine the alpha-, beta-, and gamma-evidence have been studied [5]. The most effective found thus far appears to be Dempster-Shafer [20]. Other methods included geometric mean, product, mean and maximum. This approach takes advantage of the very rough approximation to probability values, as discussed above.

LeRP Algorithm

The incremental creation of a node-to-node mapping relies critically on the correctness of the initial assignment (a 'mapping seed'). Otherwise, some kind of backtracking [9] would typically be required. The ability to find a good initial guess is a key feature of the LeRP approach. The large comparison horizons (**R**) make this possible. The find_best_beta() function is described in Figure 2. It computes the similarity measure used to help determine the 'mapping seed'. It uses a best-case approach to collecting beta-type evidence. For a given pair of nodes $\mathbf{g_i}$ and $\mathbf{g_k}$, the routine finds the best possible mapping (in terms of the **compare()** function) between each incident edge, $\mathbf{e_{ij}}$ and $\mathbf{e_{kl}}$.

Function: *find_best_beta*(G,H,A^r,B^r)

```
a. For each node g_i
b.   For each node h_k
      i.  For each edge e_ij
     ii.    For each edge e_kl
             1. beta = compare (i,j,k,l)
             2. Save beta_peak[i][k]=beta if maximal for nodes i,k
    iii.    Next l
     iv.  Next j
c.   Next k
d. Next i
e. Return beta_peak[][]
```

Figure 2. The find_best_beta() function described in pseudo-code. This function finds the best-case evidence that edges in graphs G and H form an isomorphic pairing.

```
Input:      Graph G with nodes g_i, 0<=i<N_G
            Graph H with nodes h_k, 0<=k<N_H
Output:     Mapping m(), that gives h_k = m(g_i)
            The presence of either a graph or subgraph isomorphism is
            also determined.

Steps:
1. Compute powers of adjacency matrices A^R and B^R for graphs G and H
2. beta_peak[][] = find_best_beta(G,H,A^r,B^r)
3. Clear node-to-node mappings
4. For each L, 0<=L<minimum(N_G,N_H)
      a. Let peak = 0
      b. For each unmapped node g_i
      c.   For each unmapped node h_k
            i.  Verify consistency of mapping g_i to h_k given current m()
           ii.  rho = 0
          iii.  For each mapped edge e_ij
                   1. lookup associated edge e_kl where l=m(j)
                   2. beta = compare(i,j,k,l)
                   3. gamma = compare(j,j,l,l)
                   4. rho = 1 - (1-rho)(1-beta)(1-gamma)
           iv.  Next j
            v.  alpha = compare(i,i,k,k)
           vi.  rho = 1 - (1-rho)(1- alpha)(1- beta_peak[i][k])
          vii.  If rho>peak Then
                   1. g_peak=i
                   2. h_peak =k
                   3. peak=rho
        viii.  End If
      d.   Next k
      e. Next i
      f. If peak=0 Then GoTo END
      g. Let m(g_peak)=h_peak
5. Next L

6. If (L=N_G) and (L=N_H) Then G is ISOMORPHIC to H
7. Else a subgraph isomorphism exists between G and H.
8. END
```

Figure 3. Pseudo-code for the LeRP algorithm. LeRP determines whether a graph or a subgraph isomorphism exists, and computes the associated node-to-node mapping. The comparison of (node or edge) color is somewhat application-specific and was omitted from the algorithm description.

Referring to Figure 3, evidence of structural similarity is combined via Dempster-Shafer. This is shown in steps 3.c.iii.4 and 3.c.vi. Dempster-Shafer is used to combine the various probability estimates, found via the compare() function.

Steps 3-5 in the main algorithm were repeated exactly 3 times to improve the size of the matched subgraph. On the repeated operations, an initial mapping was defined by selecting nodes from the previous matched subgraphs. Node pairs were chosen having the highest number of circuits in agreement.

There are some opportunities to reduce the compute effort described in the simplistic presentation above. For example, the values of **gamma** can be saved rather than recomputed, with only a linear cost in memory.

Memory and Compute Bounds

For the purpose of determining bounds on the needed computations and memory for LeRP, we assume the number of nodes in both input graphs equals **N**. We also assume an average degree of **D** for both graphs. **R** is a parameter, as described previously. A sparse matrix representation was used for the $\mathbf{A^r}$ matrices.

Computing the $\mathbf{N^2}$ entries of $\mathbf{A^r}$ (for some value of **r**) requires $\mathbf{N^2D}$ effort, with a sparse matrix representation. Hence the total effort to find $\mathbf{A^R}$ is on the order $\boldsymbol{O}(\mathbf{N^2DR})$. The other preliminary step, # 2, requires on the order $\boldsymbol{O}(\mathbf{N^2D^2R})$ to find the best beta array. The deepest level of nesting in the main section of the algorithm occurs in Step 4. This is in loops 4., 4.b, 4.c, and 4.c.iv. This section of the algorithm is most costly, requiring on the order $\boldsymbol{O}(\mathbf{N^3DR})$ effort.

The memory for the best-beta array is more costly than other data structures, requiring $\boldsymbol{O}(\mathbf{N^2})$. The $\mathbf{A^1}$, $\mathbf{A^2}$, $\mathbf{A^3}$ … $\mathbf{A^R}$ arrays can be stored with $\boldsymbol{O}(\mathbf{NDR})$ memory, provided a sparse representation is used. Also, a tradeoff between speed and memory is possible in LeRP. Precomputation of the best-beta array can be skipped, and the associated values simply recomputed when needed. This reduces the memory requirements to $\boldsymbol{O}(\mathbf{NDR})$.

Testing Methodology for Simulated Trials

The goal of testing was not only to verify correct operation, but also to demonstrate robustness and computational speed under various input conditions. In particular the gradual degradation in the (mean) number of nodes in matched subgraphs will be demonstrated.

Style of Graphs
Test cases were generated randomly. Graphs were created using Model A in some tests. Strongly regular (random) graphs were created for other trials. In Model A, the existence of each edge is based on an independent random variable. Model A is common in theoretical studies [26] and in reported algorithms [8][16]. All node and edge coloring was discrete in nature. The effect on performance due to the dynamic range of (randomly generated) colors was studied extensively. Dynamic ranges of color varied from one (no variation) to six. These colors might represent different types of circuit elements, different types of atoms in a molecule, or the output of a shape classifier, for example.

Style of Structural Errors
Two kinds of structural errors (or 'structural differences') were studied: errors in edges, and errors in nodes. Toggling the adjacency state of a randomly selected pair of nodes created errors in edges. When an error in a node was introduced, the entire row (and column) of the adjacency matrix was cleared. In other words the node, together with all of its edges, was eliminated.

Random structural errors can, inadvertently, result in a node having zero degree. When this occurred, the associated test trial was dropped. This is because LeRP does not process nodes with zero degree, as these have virtually no chance of being correctly matched. Hence these cases were eliminated. These cases were rare, but did occasionally happen in all the 32,000 reported trials.

Testing and Verification Procedure
Table 1 describes the test and verification procedure. The verification in step 7 is done by verifying that each entry of $\mathbf{a_{ij}^{(1)}}$ was identical to $\mathbf{b_{kl}^{(1)}}$ under the mapping **k=m(i)** and **l=m(j)**. This checks both edges and non-edges. Checking the similarity of the adjacency matrices eliminates the possibility of a false-positive result. Note that the mapping will not always contain **N** nodes, in the case of subgraph matches, for example.

Step	Procedure
1	Randomly generate graph **G** with **N** nodes using Model A (or other method).
2	Randomly generate colors for **G** using a given dynamic range.
3	Copy **G** to form an initial version of **H**.
4	Perform **2N** random interchanges of node vertices in **H**.
5	Introduce errors in **H**, either in color or structure.
6	Run the LeRP algorithm on **G** and **H**, yielding the mapping **m()**.
7	Verify the mapping **m()** preserves adjacencies.

Table 1. Test and verification procedure. Over 32,000 test trials were processed in all. A single trial consisted of the above steps. This procedure automated testing, permitting a large number of trials to be generated, and enabled the study of different parameters and performance metrics. Verification in step 7 ensured that the reordered versions of the adjacency matrices were identical.

An algorithm by McKay [17] was also used to verify node-to-node mappings from LeRP. Mappings were found to be identical in [15]. This was done in a separate test procedure (not as in Table 1) and it was performed for the graph-matching case only. In other words, these tests were performed on trials with no structural errors. The goal of using McKay's algorithm was simply to verify the fundamental operation of LeRP, via an independent check.

Testing Results for Simulated Trials

An important objective in testing was to quantify how quickly matching subgraphs could be computed, and to determine the dynamic range of coloring needed to yield an acceptable number of nodes in a matched subgraph. These two metrics can help quantify the utility of LeRP for an application.

Measuring Tolerance to Structural Errors

The tolerance to structural differences in input graphs (or 'structural errors') was studied for two different cases: errors in nodes and errors in edges. Performance is quantified by the percentage of nodes appearing in the matched subgraph. The effect of coloring is also included in this study.

Figure 4 demonstrates tolerance to structural errors in nodes. The plot contains contour lines that show the (mean) percentage of nodes in the matched subgraph. The horizontal axis gives the percentage of nodes that were removed from one of the two input graphs (along with incident edges). The vertical axis indicates the dynamic range of discrete color values that were randomly generated for nodes and edges.

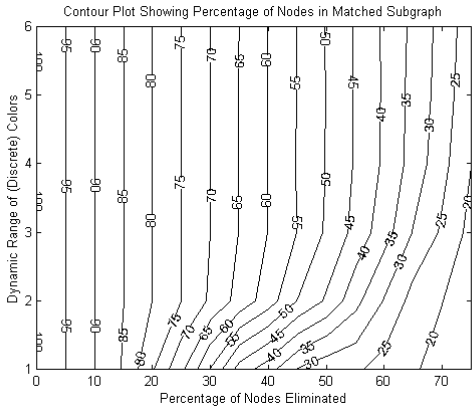


Figure 4. Contour lines indicate the percentage of nodes that remain in matched subgraphs when structural errors are added to nodes. Results are near maximal with a very modest dynamic range of node and edge coloring (four). Plot summarizes 9600 trials involving 100 node graphs. Color values are discrete. Model A graphs were used with a 10% probability of an edge.

Referring to Figure 4, the '90' contour line is an example of an ideal result. This shows that when 10% of the nodes were dropped, 90% remained in the match. Because the line is completely vertical, it means that node and edge coloring was not necessary in order to maintain the 90% match.

When 20% of the nodes are removed, some coloring is needed to aide the matching process in achieving maximal results. In this case a dynamic range of only two was required to produce an ideal result for the average case. In trials with 30%-50% corruption a dynamic range of approximately three was needed.

The above results are considered to be quite good. These indicate that *very little* dynamic range in node and edge coloring is needed in order to obtain near-maximal matching performance. Similar results follow in the case for structural errors in edges. Here, the adjacency state was randomly toggled.

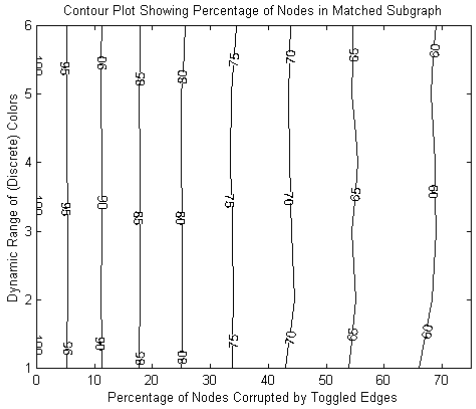
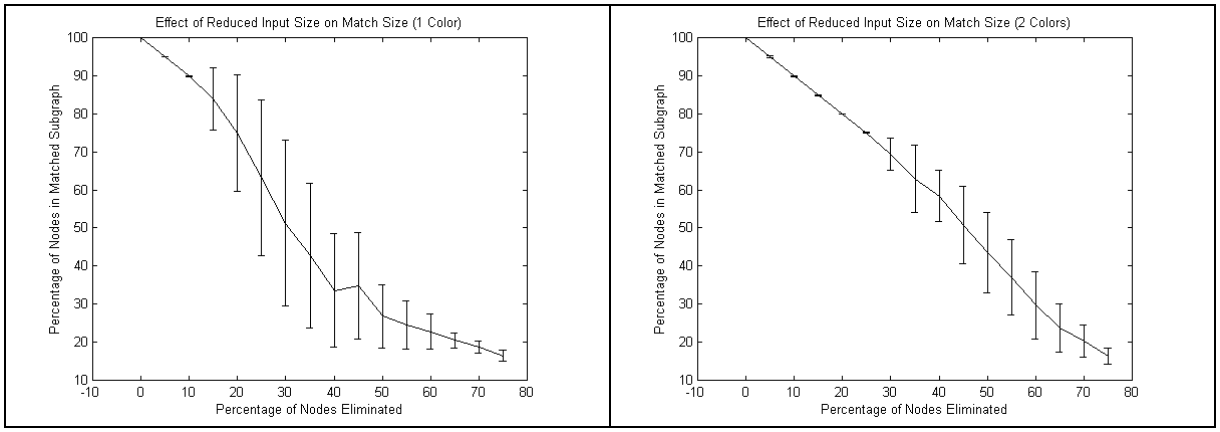
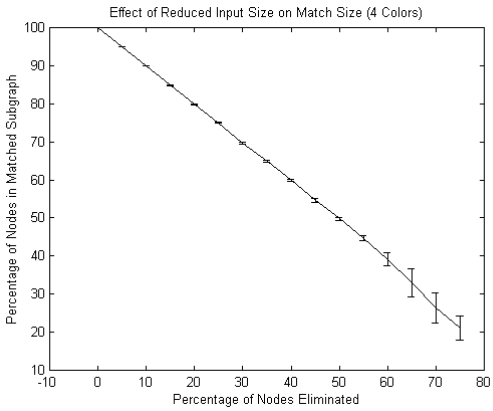


Figure 5. Contour lines indicate the percentage of nodes that remain in matched subgraphs when structural errors are added to edges. Plot summarizes 9600 trials involving 100 node graphs. Color values are discrete. Model A graphs were used with a 10% probability of an edge. The plot indicates that this type of error can be tolerated with a low dynamic range of node and edge coloring (four).

Figure 5 demonstrates the tolerance of LeRP to random structural errors in edges. Mean sizes reported in the figure are better than the case for errors in nodes. This is to be expected. Consider that in some cases two randomly added edges will be adjacent to the same node. So, if that one node were eliminated from the output subgraph, then so would both of the extra edges. Hence, eliminating one node from the subgraph mapping can take care of two (or more) structural errors. This is why (referring to Figure 5) a corruption level of 10% can result in a match size that is slightly higher than 90%, on average.

To further illustrate performance in terms of subgraph size, Figures 6-8 show results for a dynamic range of 1, 2 and 4 colors individually. These plots also include the standard deviation for each set of 100 trials.





Figures 6-8. Mean and standard deviation of the percentage of nodes remaining in matched subgraphs when structural errors are added to nodes. Results indicate near maximal performance with up to 50% corruption and a dynamic range of 4 colors. Near maximal results are possible with up to 25% corruption if two colors are present. Each error bar describes the standard deviation a set of 100 trials.

Figure 9 demonstrated the improved performance that can be realized with increased coloring (dynamic ranges of 1,2, 3, and 4).

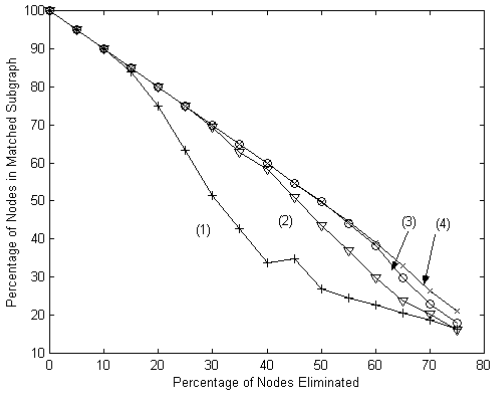


Figure 9. Effect of reduced input size on (mean) size of matched subgraph for 1,2,3 and 4 colors. Structural errors were added to nodes. Results show near maximal performance as the dynamic range of color increases. This is indicated by the asymptotic approach of the curves to a line with slope of negative one. A total of 6400 trials are summarized in the plot.

Figure 9 shows the mean number of nodes in the output subgraph. Note the asymptotic approach to a line with slope of negative one, as the dynamic range of color increases. This line represents the maximal subgraph case where removing a single node from the input results in a single node being eliminated from the output. This result is considered to be quite good, as it indicates a very gradual degradation in the size of the computed subgraph, with minimal dynamic range of node and edge color.

Measuring Tolerance to Coloring Errors

Next we document the effect of errors in graph coloring on the number of nodes in matched subgraphs. In Figure 10, errors are introduced into both node and edge colors, randomly.

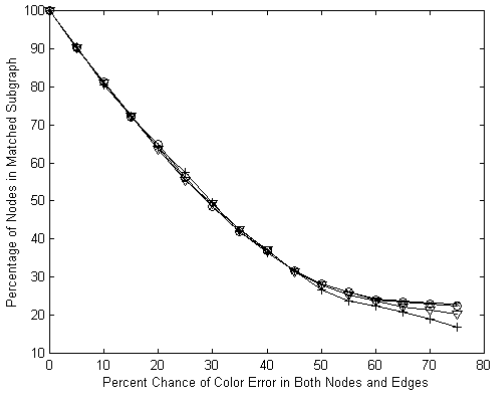


Figure 10. Effect of coloring errors on (mean) size of matched subgraph for 1,2,3 and 4 colors. Coloring errors were added to both nodes and edges. Results are slightly above a line with slope –2, indicating a good result for these two types of noise. A total of 6400 trials are summarized in the plot.

Figure 10 demonstrates relatively consistent performance, irrespective of the dynamic range of color. The mean subgraph size tapers off at a rate that is steeper than the slope of –1 seen previously (the above is between –1 and –2 in slope). This is because the effect of corrupting both nodes and edges is accumulative. Poisoning node colors tends to eliminate nodes on a one-for-one basis (as described previously). However, random errors in edge colors have a more subtle effect on the output size (as was also seen with the structural errors in edges).

Benchmarks of Execution Time

LeRP processing time is shown in Figure 11, on a log scale. The processing time includes all the time needed to process the pair of input graphs – but not the time needed to generate the random graphs. Results are shown for graphs containing no coloring and no structural errors. Tests were run on a Sun Enterprise™ 450 server with 400-MHz UltraSPARC™-II processors, and 1GB RAM, running Solaris 2.7. Note, tests on a Windows ® PC with 550 MHz Duron™ processor and 256 MB RAM required very nearly the same processing time.

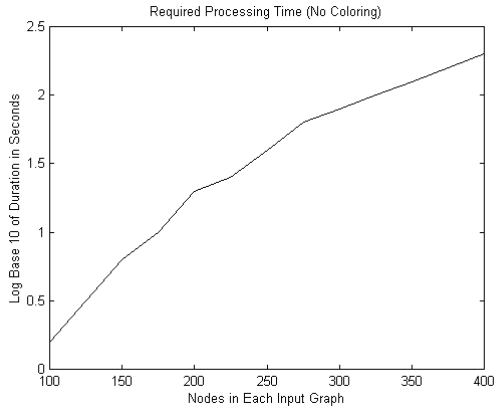


Figure 11. Mean processing time for graphs of varying size with no coloring. No structural errors were introduced. The 400 node graphs required a mean processing time of 3 minutes, with a standard deviation of 5 seconds (3%). The error bars were dropped, as they would not have been visible on the log plot with such low standard deviation. Graphs with 100 nodes took approximately 1.5 seconds. Coloring improves speed, for example 100 node graphs with a dynamic range of four colors require ~0.4 seconds.

Effect of Reduced Dynamic Range of Node Color

Table 2 summarizes tests that approximate conditions reported by Messmer and Bunke [16]. These tests examine the effect of the dynamic range of node color on the size of the matched subgraph. Edges had no coloring in these tests.

We use Model A to generate graphs, with probability 0.15 for edges, to approximate the density of edges reported in [16]. Results indicate that LeRP can maintain a near maximal size for the matched subgraph, provided the dynamic range of (discrete) node coloring stays at or above 10% of the number of nodes. As described previously, Messmer and Bunke [16] report degraded results with dynamic a range below 20%.

Dynamic Range Of Node Color	Size of Matched Subgraph (Mean +/- Standard Deviation)
20	50 +/- 0.1
15	50 +/- 0.1
10	50 +/- 0.2
7	49 +/- 5.2
5	47 +/- 9.3

Table 2. Effect of dynamic range of node color on size of matched subgraph. Edges had no coloring in these tests. Trials used graphs with 100 nodes, and eliminated 50 nodes from one of the inputs, making the ideal size of the matched subgraph, above, equal to 50 nodes. Each trial (row) consisted of 100 randomly generated tests. Model A was used to generate graphs with a probability = 0.15. Results show that LeRP can maintain near maximal matches, provided the node coloring stays above 10% of the number of nodes.

Effect of Increased Density of Edges

Table 3 summarizes tests that examine the effect of an increased density of edges on the size of the matched subgraph. Results show that denser graphs are more challenging for LeRP, but can be accommodated given a slight increase in the dynamic range of node and edge coloring.

Model A Probability	Dynamic Range of Node and Edge Color	Size of Matched Subgraph (Mean +/- Standard Deviation)
0.1	4	50 +/- 0.5
0.2	4	50 +/- 3.6
0.3	4	46 +/- 11
0.4	4	44 +/- 15
0.5	4	38 +/- 19
0.1	5	50 +/- 0.5
0.2	5	50 +/- 0.0
0.3	5	50 +/- 3.1
0.4	5	49 +/- 6.0
0.5	5	49 +/- 6.8

Table 3. Effect of increased density of edges on size of matched subgraph. Model A was used in each case. The dynamic range of node and edge coloring is also shown. Test conditions were similar to Table 2. Results indicate that a slight increase in the dynamic range of coloring can accommodate the denser graphs.

Tests with Strongly Regular Graphs

Table 4 summarizes tests involving strongly regular graphs. Here, the size of the matched subgraph was examined as the degree of the graphs was increased. This style of testing was included because regular graphs are the most challenging type for isomorphism algorithms [33]. This makes for a difficult matching problem for the subgraph case. Here, the local comparisons (**r=1**) yield little distinction due to the near identical degree. Furthermore, the more distance comparisons (**r~N**) tend to be corrupted by structural errors. This makes for a challenging style of graph for the structural comparisons.

Test conditions are similar to Tables 2 and 3, in that the maximal subgraph had 50 nodes. Because the input graphs were random in nature, the degree was not exactly the same for every node. The actual mean degree is reported in the test results.

Results demonstrate that an increasing degree does become more challenging for LeRP. However, as with the increased density of Table 3, this can be accommodated with a modest increase in the dynamic range of node and edge coloring.

Target Degree	Actual Mean Degree	Dynamic Range of Node and Edge Color	Size of Matched Subgraph (Mean +/- Standard Deviation)
10	9.9	4	50 +/- 0.2
20	19.9	4	50 +/- 0.0
30	29.6	4	44 +/- 15
10	9.9	5	50 +/- 0.2
20	19.9	5	50 +/- 0.0
30	29.7	5	50 +/- 0.0

Table 4. Size of matched subgraph for strongly regular graphs. Increasing the degree makes matching more challenging for the LeRP algorithm. But, as with the denser graphs in Table 3, the drop in performance can be recovered with only a slight increase in the dynamic range of node and edge coloring. Test conditions were similar to Table 2, but with the strongly regular graphs.

Conclusion and Areas for Further Investigation

The objective of this study was to develop an approximate subgraph isomorphism algorithm that could work well in a variety of applications. LeRP is successful in this regard, with worst-case computational requirements on the order $\mathbf{O(N^3D^2R)}$, where **N** is the number of nodes, **D** is the mean degree, and **R** is the highest power of the adjacency matrix used in processing. The algorithm has demonstrated a high tolerance to structural and coloring errors, and also has reasonable memory requirements. We have also demonstrated that LeRP has a low dependence on the dynamic range of coloring. The key to the success of the LeRP algorithm is a dynamic comparison horizon for measuring structural similarity. This permits a mapping to be determined via a deterministic amount of processing.

LeRP is an improvement on previous work by DePiero [5]. Compared to the previous work, LeRP is greatly simplified and significantly more efficient. For example, previous tests were limited to graphs with 40 nodes, where tests here ran up to 400 nodes, with reasonable execution times. Future studies with LeRP include applications in the areas of range image registration, and electronic circuit comparisons. Studies involving the necessary limits on the parameter **R**, alternate methods of computing node similarity, and of combining evidence, are also under consideration.

References

1. H.G. Barrow, R.M. Burstall, Subgraph Isomorphism, Matching relational structures and maximal cliques, Information Processing Letters, 4 (1976) 83-84.
2. R.C. Bolles, R.A. Cain, Recognizing and locating partially visible objects: the local-feature-focus method, Int. J. of Robotics Research, 1 (3) (1982) 1236-1253
3. H. Bunke, G. Allerman, Inexact graph matching for structural pattern recognition, Pattern Rec. Letters, 1 (4) (1983) 245-253
4. M.S. Costa, L.G. Shapiro, Scene analysis using appearance-based models and relational indexing, IEEE Symposium on Computer Vision, Coral Gables, FL, Nov. 1995, 103-108.
5. F. W. DePiero, M. M. Trivedi, S. Serbin, Graph Matching Using a Direct Classification of Node Attendance, Pattern Recognition Journal, vol. 29, no. 6, pp. 1031-1048, 1996.
6. A.D.J. Cross, E.R. Hancock, Graph matching with a dual-step EM algorithm, IEEE Trans. Pattern Analysis and Machine Intelligence, 20 (11) (1998) 1236-1253
7. M.A. Eshera, K.S. Fu, A graph distance measure for image analysis, IEEE Trans. Systems, Man and Cybernetics, 14 (3) (1984) 398-408.
8. S. Gold, A Rangarajan, A graduated assignment algorithm for graph matching, IEEE Trans. Pattern Analysis and Machine Intelligence, 18 (4) (1996) 377-388.
9. W.E.L. Grimson, Object recognition by computer, MIT Press, 1990
10. R.C. Wilson, E.R. Hancock, Structural matching by discrete relaxation, IEEE Trans. Pattern Analysis and Machine Intelligence, 19 (6) (1997) 634-648.

11. R. Myers, R.C. Wilson, E.R. Hancock, Bayesian graph edit distance, IEEE Trans. Pattern Analysis and Machine Intelligence, 22 (6) (2000) 628-635.
12. R.M Haralick, L.G. Shapiro, The consistent labeling problem I, IEEE Trans. Pattern Analysis and Machine Intelligence, 1 (1979) 173-184.
13. L.G. Shapiro, R.M Haralick, Structural descriptions and inexact matching, IEEE Trans. Pattern Analysis and Machine Intelligence, 3 (5) (1981) 504-519.
14. L.G. Shapiro, R.M Haralick, A metric for comparing relational descriptions, IEEE Trans. Pattern Analysis and Machine Intelligence, 7 (1985) 90-94.
15. D.W. Krout, LeRP: An Algorithm for Finding Subgraph Isomorphisms with Applications to VLSI, Master's Thesis, Cal Poly State University, San Luis Obispo, CA, 2001.
16. B. T. Messmer, H. Bunke, A new algorithm for error-tolerant subgraph isomorphism detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 20 (5) (1998) 493-504.
17. B. McKay. Practical Graph Isomorphism, Congressus Numerantium, 30 (1981) 45-87.
18. N.J. Nillson, Principles of Artificial Intelligence, Tioga, Palo Alto, CA, 1980.
19. A. Sanfeliu, K.S. Fu, A distance measure between attributed relational graphs for pattern recognition, IEEE Trans. Systems, Man and Cybernetics, 13 (1983) 353-363.
20. G. Shafer, A Mathematical Theory of Evidence, Princeton University Press, Princeton, New Jersey, 1976.
21. G. Stockman, Object recognition and localization via pose clustering, Computer Vision, Graphics and Image Processing, 40 (1987) 361-387.
22. Y. Lamden, H. Wolfson, Geometric hashing: a general and efficient model-based recognition scheme, Proc. 2nd Int. Conf. On Computer Vision, Tarpon Springs, FL (Nov. 1988) 238-249.
23. E.K. Wong, Three-dimensional object recognition by attributed graphs, H. Bunke and A. Sanfeliu, eds, Syntactic and Structural Pattern Recognition – Theory and Applications, World Scientific (1990) 381-414.
24. A. Rosenfeld, R. Hummel, S. Zucker, Scene labeling by relaxation operators, IEEE Trans. Systems, Man and Cybernetics, 6 (1976) 420-453.
25. R. Hummel, S. Zucker, On the foundations of relaxation labeling processes, IEEE Trans. Pattern Analysis and Machine Intelligence, 5 (1983) 267-287.
26. E. M. Palmer, Graphical Evolution – An Introduction to the Theory of Random Graphs, Wiley-Interscience, 1985.
27. W J Christmas, J Kittler and M Petrou, 1995. ``Structural matching in Computer Vision using Probabilistic Relaxation". IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-17, pp 749--764.
28. W J Christmas, J Kittler and M Petrou, 1996. ``Probabilistic feature-labeling schemes: modeling compatibility coefficient distributions". Image and Vision Computing, Vol 14, pp 617- 625.
28. W J Christmas, J Kittler and M Petrou, 1996. ``Labeling 2-D geometric primitives using probabilistic relaxation: reducing the computational requirements". Electronic Letters, Vol 32(4), pp 312--314.
30. J Kittler, M Petrou and W J Christmas, 1998. ``Non-iterative contextual correspondence matching". Pattern Recognition Journal, Vol 31, No 10, pp 1455--1468.
31. L.R. Foulds, Graph Theory Applications, Springer-Verlag, New York, 1992.
32. D.B. West, Introduction to Graph Theory, 2nd ed., Prentice-Hall, Upper Saddle River, New Jersey, 2001.
33. R. C. Read and D. G. Corneil, The graph isomorphism disease, Journal of Graph Theory, 1 (1) 339-363 (1977).
34. B. Luo, E. R. Hancock, Structural Graph Matching Using the EM Algorithm and Singular Value Decomposition, IEEE Trans. Pattern Analysis and Machine Intelligence, 23 (10) 1120-1136.